ECLIPSE AND EMBEDDED LINUX DEVELOPERS: WHAT IT CAN AND CANNOT DO FOR YOU

Anna Dushistova
anna.dushistova@gmail.com
Why IDE?

From wikipedia:

“...IDEs are designed to maximize programmer productivity by providing tightly-knit components with similar user interfaces. This should mean that the programmer has much less mode switching to do than when using discrete development programs...”
TYPICAL CONTENT OF AN IDE

- a source code editor
- a compiler and/or an interpreter
- build automation tools
- a debugger
- SCM integration
About Eclipse

What is it?

- Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle.

History of Eclipse


- On Feb 2, 2004 the Eclipse Board of Stewards announced Eclipse’s reorganization into a not-for-profit corporation. The founding Strategic Developers and Strategic Consumers were Ericsson, HP, IBM, Intel, MontaVista Software, QNX, SAP and Serena Software.
WHAT CAN ECLIPSE OFFER?

- C/C++ Development Tooling (CDT) for edit/compile/debug cycle
- Target Management (TM) or Target Communication Framework (TCF) with Target Explorer (TE) for working with remote hosts
- CVS, SVN and Git integration
- Linux Tools for various Linux specific performance monitoring tools (gprof, lttng, oprofile, systemTap)
SOME COMPANIES BEHIND THESE PROJECTS

ERICSSON

BlackBerry

Google

intel

IBM

XILINX

mentor embedded

freescale

polytechnique montréal
But...

- Is it possible to use Eclipse with standard versions of these plugins for embedded Linux development?
  - Install additional Target Management feature(SCP file subsystem) (update site: http://download.eclipse.org/tm/updates/3.4/)
  - Install latest version of Target Explorer (update site: http://download.eclipse.org/tools/tcf/builds/development/nightly/)
WORKING WITH C/C++ CODE

1. Code Analysis
   Framework “Codan”

   1. Now supports external code analysis tools like cppcheck:
      - External tools can be configured using Codan’s preference page
      - External tools are invoked automatically when a C/C++ file is saved
      - The output of these tools can be displayed as editor markers
Building Your Application

- Works perfectly for a local toolchain out of the box!
- However, when it comes to using cross-compilation tools...
## Build Systems Supported in CDT

<table>
<thead>
<tr>
<th>CDT Managed Make</th>
<th>Standard Make</th>
<th>Autotools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makefiles are generated by CDT</td>
<td>You write your own makefiles</td>
<td>Build scripts are generated by automake tools</td>
</tr>
<tr>
<td>Doesn’t have support outside CDT</td>
<td>Standard Linux way of software development</td>
<td>Standard Linux way of software development</td>
</tr>
<tr>
<td>Requires complex toolchain definition for CDT, has a lot of limitations</td>
<td>Needs only a set of environment variables</td>
<td>Needs only a set of environment variables and proper arguments to pass to autoconf</td>
</tr>
</tbody>
</table>
DEALING WITH CDT’S MANAGED MAKE

- How can we integrate CDT with our cross-compilation tools?
  - Write a plugin for Eclipse that adds support for it
  - Use “Cross GCC”
  - Redefine settings for each project manually
Defining your own toolchain for CDT

- If you don’t need any special settings, just extend existing definitions for “Linux GCC”
- The more special your needs, the more changes you will have to make
- Changes are not obvious and require some Eclipse experience

```
<targetPlatform
    id="cdttest309024.gnu.platform.base"
    name="Debug"
    binaryParser="org.eclipse.cdt.core.ELF"
    osList="linux,hpux,aix,qnx"
    archList="all">
</targetPlatform>
<builder
    superClass="cdt.managedbuild.target.gnu.builder"
    id="cdt.test309024.builder.base">
</builder>
<tool command="$CCX"
    id="cdt.test309024.cpp.compiler" isAbstract="false"
    name="Anna's GNU G++ Compiler" superClass="cdt.managedbuild.tool.gnu.cpp.compiler">
</tool>
```
**USING “CROSS GCC”**

- Requires just two settings:
  - path
  - prefix (make sure you add dash in the end)
REDEFINING SETTINGS MANUALLY

- Must be done for each configuration in each project in use
- Requires the following modifications of project properties:
  - changing gcc to your <arch>-gcc in Settings for each tool
  - Modifying PATH variable (adding your cross-toolchain path)
Since summer 2012 there are two options:

- Remote System Explorer (RSE) provided by Target Management project
- Target Explorer (TE) coming from Target Communication Framework project
# Remote Systems Explorer vs. Target Explorer

<table>
<thead>
<tr>
<th>Remote Systems Explorer</th>
<th>Target Explorer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mature, stable project (1.0 in November 2006)</td>
<td>1.0 version was released in June 2012, designed to be a lightweight successor of RSE for embedded</td>
</tr>
<tr>
<td>Transport protocol agnostic</td>
<td>Centered around Target Communication Framework (TCF) protocol and assumes there is a TCF agent on the remote target</td>
</tr>
<tr>
<td>Does not support auto-discovery</td>
<td>Supports auto-discovery of connections and services</td>
</tr>
</tbody>
</table>
Vendor-neutral, lightweight, extensible network protocol mainly for communicating with embedded systems (targets)

- Has a transport-agnostic channel abstraction
- Supports auto-discovery of targets and services
RUNNING YOUR APPLICATIONS USING RSE

- Use Remote Systems Explorer to copy your application on to the target and then a terminal to launch it
- Use Remote Launch to launch your application from Eclipse
- Transferring more than one file (e.g. application and library) is not supported in the Remote Launch
Running Your Applications Using TE

- Remote Launch from Target Explorer allows you to transfer all necessary files and run your application in one step.
- Output is shown in ANSI terminal.
Options offered:

- GDB/GDB server solution
  - Automatic remote debugger (launches gdbserver on remote side automatically, doesn’t support attaching to a running process)
  - Current implementation uses RSE
  - TE-based solution pending review
- Manual remote debugger (you have to launch gdbserver manually, supports attach)
- TCF-agent-based debugger
  - only x86 debugging support in the open source TCF agent at the moment
DEBUGGING YOUR APPLICATIONS-GDB

Tuesday, November 6, 2012

```
Missing separate debuginfo for /lib/libc.so.6
Try: zypper install -C "debuginfo(build-id)=ee302691046515fe3766ae3b7d47af3e3a8d063"
```

```
Num Type Disp Enb Address What
1 breakpoint keep y 0x0804841d in main at ../src/hello1.c:15
breakpoint already hit 1 time
```
Debugging your Applications - TCF

```c
static CPU_t *smcpu = NULL;

// whoa first time, gotta prime the pump...
if (p_table) {
    p_table = procs_refresh(NULL, Frames_libflags);
    putp(Cap_clr_scr);
    putp(Cap_rmam);
}
```

```
8 root  RT  0  0  0  0  S  0.0  0.0  0:00.00 migration/0
9 root  0.20  0  0  0  0  S  0.0  0.0  0:00.00 cpuset
10 root  0.20  0  0  0  0  S  0.0  0.0  0:00.00 khelper
11 root  0.20  0  0  0  0  S  0.0  0.0  0:00.28 netsns
12 root  20  0  0  0  0  S  0.0  0.0  0:00.00 kworker/u:1
312 root  0.20  0  0  0  0  S  0.0  0.0  0:00.00 kblockd
```
Linux Measurement and Diagnostic Tools

- **Profiling**
  - oprofile: remote launching is not supported, only local
  - perf: remote launch exists, but uses Remote Development Tools instead of RSE or TE
  - valgrind: only local launch is supported

- **Tracing**
  - Supports LTTng 2.0, uses RSE for connecting to remote targets
Yocto Eclipse plugin (learn more at the Yocto Developer day on Thursday!)

Technology is there!

But:

- almost every piece requires non-trivial actions to make it work the way we need
- some cannot be used for cross development
- pieces do not integrate very well

Overall, the set of plugins and features we installed can hardly be called an IDE.
Demo Setup

- Host:
  - Yocto standalone cross toolchain 1.2.1 for x86(32 bit)
  - Eclipse IDE for C/C++ Developers package Juno SR1

- Target(qemu x86):
  - Stable kernel 3.6.2 from kernel.org
  - Yocto core-image-sato-sdk filesystem image for x86 qemu
  - Lttng 2.0 packages(lttng-modules-2.0.3, lttng-tools-2.0.1, lttng-ust-2.0.3 and userspace-rcu-0.7.3)
  - Latest TCF agent
QUESTIONS?
ACKNOWLEDGEMENTS

• Uwe Stieber
• Laurette Wharton
• Stanislav Yakovlev
• Eclipse committers and contributors
• Yocto project
THANK YOU!